포팅 메뉴얼

1. 개발환경

1.1 Frontend

Next.js 및 관련 라이브러리

- next : 14.2.16
- react : 18
- react-dom : 18
- zustand : 5
- react-modal : 3.16.1

캘린더 서비스

- fullcalendar : 6.1.15

드래그 앤 드롭

- hello-pangea/dnd : 17.0.0

PDF 생성

- jspdf : 2.5.2

파일 업로드

- react-dropzone : 14.3.5

1.2 Backend

자바

- Java OpenJDK 17

Spring Boot 스타터

- spring-boot-starter-actuator
- spring-boot-starter-data-jdbc
- spring-boot-starter-data-jpa
- spring-boot-starter-data-redis
- spring-boot-starter-oauth2-client
- spring-boot-starter-security
- spring-boot-starter-web
- spring-boot-starter-websocket

개발 편의성 도구

- lombok
- spring-boot-devtools

데이터베이스 관련

- mysql-connector-j

테스트 관련

- spring-boot-starter-test
- spring-security-test
- junit-platform-launcher

API 문서화

- springdoc-openapi-starter-webmvc-ui:2.6.0

JWT (JSON Web Token)

- jjwt-api:0.12.5
- jjwt-impl:0.12.5
- jjwt-jackson:0.12.5

AWS S3 연동

- spring-cloud-starter-aws:2.2.6.RELEASE
- aws-java-sdk-s3:1.12.287
- s3:2.20.0

1.3 Server

- Ubuntu 20.04.6 LTS
- Nginx 1.18.0
- Docker 24.0.7
- Docker Compose 2.30.1
- Jenkins 2.483

1.4 Database

- MySQL 8.0.39
- Redis

1.5 IDE

- Visual Studio Code

- IntelliJ IDEA

1.6 형상 / 이슈관리

- Gitlab
- Jira

1.7 기타 툴

- Postman
- Swagger

2. 환경 변수

2.1 Frontend

> NEXT_PUBLIC_OPENAI_API_KEY

> NEXT_PUBLIC_SERVER_URL

2.2 Backend

application.yml 을 작성하여 환경변수를 관리

```
spring:
  datasource:
    url: ${SQL_DATA_URL} #jdbc:mysql://localhost:3306/mydb
    username: ${SQL_USER_NAME}                          # MySQL
    password: ${SQL_USER_PASSWORD}                    # MySQL 비밀
    driver-class-name: com.mysql.cj.jdbc.Driver
  data:
    redis:
```

```
      host: ${REDIS_HOST}  # Redis 호스트
      port: ${REDIS_PORT}  # Redis 포트
      password: ${REDIS_PASSWORD}  # Redis 비밀번호 설정
      timeout: 6000    # 연결 타임아웃 (밀리초)
      lettuce:
        pool:
          max-active: 10  # 최대 활성 커넥션 수
          max-idle: 5     # 최대 유휴 커넥션 수
          min-idle: 1     # 최소 유휴 커넥션 수
  h2:
    console:
      enabled: true
  jpa:
    hibernate:
      ddl-auto: update
      # create: 애플리케이션 시작 시 테이블 생성
      # create-drop: 애플리케이션 종료 시 테이블 삭제
      # update: 애플리케이션 시작 시 테이블이 없으면 생성, 있으면 유지
    show-sql: true
    open-in-view: false

# swagger 설정
springdoc:
  default-consumes-media-type: application/json;charset=UTF-8
  api-docs:
    enabled: true
    path: '/v3/api-docs'
  swagger-ui:
    enabled: true
    path: '/swagger-ui.html'
    try-it-out-enabled: true
    operations-sorter: alpha

cloud:
  aws:
    s3:
      bucket: ${S3_BUCKET}
    region:
```

```
      static-region: ${S3_REGION}
    credentials:
      accessKey: ${S3_ACCESS_KEY}
      secretKey: ${S3_SECRET_KEY}

jwt:
  secret-key: ${SECRET_KEY}
  access-token-exp: ${ACCESS_TOKEN_EXP}
  refresh-token-exp: ${REFRESH_TOKEN_EXP}
openai:
    api-key: ${OPENAI_API_KEY}
    api-url: ${OPENAI_API_URL}
    api-model: ${OPENAI_API_MODEL}
    system-prompt: "당신은 JIRA 전문가입니다. 사용자가 JIRA와 관련된
```

3. EC2 세팅

3.1 Docker & Compose 설치

```
# 패키지 업데이트
sudo apt-get update
sudo apt-get upgrade -y

# 필요한 패키지 설치
sudo apt-get install -y ca-certificates curl gnupg

# Docker GPG 키 추가
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud

# Docker 저장소 추가
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/ke
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d
```

```
# 패키지 캐시 업데이트
sudo apt-get update

# Docker 엔진 설치
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

# Docker 서비스 활성화
sudo systemctl enable docker

# Docker 서비스 시작
sudo systemctl start docker

# Docker 상태 확인
sudo systemctl status docker

# Docker 권한 주기
sudo usermod -aG docker $USER

# 설치 확인
docker --version
docker run hello-world

# Docker Compose 설치
sudo curl -L "https://github.com/docker/compose/releases/late
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

3.2 Nginx 설치 및 Reverse Proxy 세팅

3.2.1 Nginx 설치

```
# 패키지 업데이트
sudo apt-get update
sudo apt-get upgrade -y

# Nginx 설치
sudo apt-get install -y nginx
```

```
# Nginx 시작
sudo systemctl start nginx

# Nginx를 부팅 시 자동 시작하도록 설정
sudo systemctl enable nginx

# Nginx 상태 확인
sudo systemctl status nginx
```

3.2.2 Nginx 파일 설정

```
# Nginx 파일 설정
sudo nano /etc/nginx/sites-available/app.conf
```

3.2.3 app.conf 파일 작성

```
server {
    listen 80;
    server_name k11a403.p.ssafy.io;
    return 301 https://$host$request_uri; # HTTP 요청을 HTTPS로

}


server {
    listen 443 ssl;
    server_name k11a403.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k11a403.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/k11a403.p.ssafy
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;



    # WebScoker 요청을 위한 설정
    location /api/ws {
```

```
        proxy_pass http://localhost:8081/ws;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
    }

    # 프론트엔드 서비스
    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto https;
    }


    # 백엔드 서비스
    location /api/ {
        rewrite ^/api/(.*) /$1 break;
        proxy_pass http://localhost:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto https;
    }


    # Swagger API Docs
    location /v3/api-docs {
        proxy_pass http://localhost:8081/v3/api-docs;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Swagger UI
```

```
    location /swagger-ui.html {
        proxy_pass http://localhost:8081/swagger-ui.html;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /dozzle/ {
        rewrite ^/dozzle(/.*)$ $1 break;
        proxy_pass http://localhost:9999;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
    }


}
```

3.3 Jenkins(Docker) 설치

```
sudo docker run -d --name jenkins \
  -p 8080:8080 -p 50000:50000 \
  -v jenkins_home:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
  jenkins/jenkins:jdk17
```

3.4 MySQL(Docker) 설치

```
docker run -d --name mysql \
  -e MYSQL_ROOT_PASSWORD=my-secret-pw \
  -e MYSQL_DATABASE=mydatabase \
  -e MYSQL_USER=myuser \
```

```
  -e MYSQL_PASSWORD=mypassword \
  -p 3306:3306 \
  -v mysql_data:/var/lib/mysql \
  mysql:latest
```

3.5 Redis(Docker)설치

```
docker run -d --name redis \
  -p 6379:6379 \
  -v redis_data:/data \
  redis:latest redis-server --save 60 1 --loglevel warning
```

3.6 EC2 Port

```
# 현재 UFW 상태 확인 (활성화/비활성화 여부 확인)
sudo ufw status

# SSH 포트(22번) 허용 (SSH 접속 보장을 위해 항상 먼저 허용)
sudo ufw allow 22

# HTTP 포트(80번) 허용 (웹 서버 접근 허용)
sudo ufw allow 80

# UFW에 추가된 규칙 확인 (inactive 상태에서도 조회 가능)
sudo ufw show added

# UFW 활성화 상태에서 규칙 목록을 번호와 함께 조회
sudo ufw status numbered

# 방화벽 활성화 (추가된 규칙 적용, SSH 연결 끊김에 주의)
sudo ufw enable

# UFW 활성화 상태에서 번호를 확인 후, 삭제할 규칙 식별
sudo ufw status numbered
```

```
# 특정 규칙 삭제 (번호를 사용하여 삭제, 예: 2번 규칙 삭제)
sudo ufw delete [번호]  # 예: sudo ufw delete 2

# 방화벽 비활성화 (모든 규칙을 중단)
sudo ufw disable
```

```
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
8989                       ALLOW       Anywhere
443                        ALLOW       Anywhere
80                         ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
6379                       ALLOW       Anywhere
3306                       ALLOW       Anywhere
9999                       ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
8989 (v6)                  ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
6379 (v6)                  ALLOW       Anywhere (v6)
3306 (v6)                  ALLOW       Anywhere (v6)
9999 (v6)                  ALLOW       Anywhere (v6)
```

4. Front, Backend 배포 Dockerfile

4.1 Frontend

```
# Node.js 20.15.1 이미지 사용
FROM node:20.15.1

# 작업 디렉토리 설정
WORKDIR /app
```

```
# 패키지 설치
COPY package*.json ./
RUN npm install

# 소스 코드 복사
COPY . .

# 환경 변수 설정
ARG NEXT_PUBLIC_SERVER_URL
ENV NEXT_PUBLIC_SERVER_URL=$NEXT_PUBLIC_SERVER_URL

# 빌드
RUN npm run build

# 포트 노출
EXPOSE 3000

# 애플리케이션 시작
CMD ["npm", "start"]
```

4.2 Backend

```
# OpenJDK 17 이미지 사용
FROM openjdk:17-jdk-slim

# 작업 디렉토리 설정
WORKDIR /app

# 시스템 패키지 업데이트 및 필요한 패키지 설치
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
    findutils \
    coreutils \
    procps && \
```

```
    rm -rf /var/lib/apt/lists/*

# Gradle Wrapper 복사
COPY gradlew ./
COPY gradle ./gradle

# 소스 코드 복사
COPY . .

# gradlew에 실행 권한 부여
RUN chmod +x gradlew

# 환경 변수 설정
ARG SPRING_PROFILES_ACTIVE
ENV SPRING_PROFILES_ACTIVE=$SPRING_PROFILES_ACTIVE

# 빌드
RUN ./gradlew clean build -x test --no-daemon

# 포트 노출
EXPOSE 8081

# 애플리케이션 시작
CMD ["java", "-jar", "build/libs/app.jar"]
```

5. Jenkins & Docker Compose 설정

5.2 Jenkins Pipeline 설정 (gitlab push하면 자동 배포)

```
pipeline {
    agent any

    environment {
        DOCKER_BUILDKIT = '1'
        SPRING_PROFILES_ACTIVE = 'prod'
```

```
        NEXT_PUBLIC_SERVER_URL = 'https://k11a403.p.ssafy.io/
    }

    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }

        stage('Build Backend') {
            steps {
                withCredentials([file(credentialsId: 'dev-be-
                    dir('server') {
                        // .env 파일을 환경 변수로 로드하여 빌드
                        sh 'export $(grep -v "^#" $ENV_FILE |
                        sh 'chmod +x gradlew'
                        sh './gradlew clean build -x test'
                    }
                }
            }
        }

        stage('Build Frontend') {
            steps {
                withCredentials([file(credentialsId: 'dev-be-
                    dir('client/aissue') {
                        sh '''
                        docker run --rm -u $(id -u):$(id -g)
                        npm install &&
                        npm run build
                        "
                        '''
                    }
                }
            }
        }
```

```
        stage('Deploy with Docker Compose') {
            steps {
                withCredentials([file(credentialsId: 'dev-be-
                    sh 'docker-compose down'
                    sh """
                    docker-compose --env-file $ENV_FILE build
                    --build-arg NEXT_PUBLIC_SERVER_URL \
                    --build-arg SPRING_PROFILES_ACTIVE=${env.
                    """
                    sh 'docker-compose --env-file $ENV_FILE u
                }
            }
        }
    }



    post {
        always {
            echo 'Pipeline finished.'
        }
        failure {
            echo 'Pipeline failed.'
        }
    }
}
```

5.2 Docker Compose 파일 설정

```
services:
  backend:
    build:
      context: ./server
```

```
      dockerfile: Dockerfile
    ports:
      - "8081:8080"
    environment:
      SPRING_DATASOURCE_URL: ${SQL_DATA_URL}
      SPRING_DATASOURCE_USERNAME: ${SQL_USER_NAME}
      SPRING_DATASOURCE_PASSWORD: ${SQL_USER_PASSWORD}
      SPRING_DATA_REDIS_HOST: ${REDIS_HOST}
      SPRING_DATA_REDIS_PORT: ${REDIS_PORT}
      SPRING_DATA_REDIS_PASSWORD: ${REDIS_PASSWORD}
      CLOUD_AWS_S3_BUCKET: ${S3_BUCKET}
      CLOUD_AWS_CREDENTIALS_ACCESSKEY: ${S3_ACCESS_KEY}
      CLOUD_AWS_CREDENTIALS_SECRETKEY: ${S3_SECRET_KEY}
      CLOUD_AWS_REGION_STATIC_REGION: ${S3_REGION}
      JWT_SECRET_KEY: ${SECRET_KEY}
      JWT_ACCESS_TOKEN_EXP: ${ACCESS_TOKEN_EXP}
      JWT_REFRESH_TOKEN_EXP: ${REFRESH_TOKEN_EXP}
      OPENAI_API_KEY : ${OPENAI_API_KEY}
      OPENAI_API_URL : ${OPENAI_API_URL}
      OPENAI_API_MODEL : ${OPENAI_API_MODEL}
    depends_on:
      - mysql
      - redis
    volumes:
      - backend_logs:/app/logs

  frontend:
    build:
      context: ./client/aissue
      dockerfile: Dockerfile
      args:
        NEXT_PUBLIC_SERVER_URL: ${NEXT_PUBLIC_SERVER_URL}
        NEXT_PUBLIC_OPENAI_API_KEY: ${NEXT_PUBLIC_OPENAI_API_
    environment:
      NEXT_PUBLIC_SERVER_URL: ${NEXT_PUBLIC_SERVER_URL}
      NEXT_PUBLIC_OPENAI_API_KEY: ${NEXT_PUBLIC_OPENAI_API_KE
    ports:
      - "3000:3000"
```

```
    depends_on:
      - backend

  redis:
    image: redis
    command: redis-server --requirepass ${REDIS_PASSWORD}
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
    restart: unless-stopped

  mysql:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: ${SQL_USER_PASSWORD}
      MYSQL_DATABASE: mydb
      MYSQL_USER: ${SQL_USER_NAME}
      MYSQL_PASSWORD: ${SQL_USER_PASSWORD}
    ports:
      - "3306:3306"
    volumes:
      - mysql_data:/var/lib/mysql
    restart: unless-stopped



volumes:
  backend_logs:
  redis_data:
  mysql_data:
```